# **Restaurant Management System (RMS)**

## **Table of Contents**

- 1. Introduction
- 2. Features
- 3. System Requirements
- 4. Installation
- 5. Project Structure
- 6. usage
- 7. Business Logic
- 8. Classes and Functionalities
- 9. JSON Files and Their Roles
- 10. Future Enhancements

## 1. Introduction:

The **Restaurant Management System (RMS)** is a Python-based project designed to streamline restaurant operations. It includes order management, menu management, billing, and table booking, following Object-Oriented Programming (OOP) principles. The system supports both admin and normal client roles, ensuring smooth management and usability.

## 2. Features:

### • Admin Panel:

- Full control over the system, including managing users, viewing analytics, and canceling any order.
- View customer orders and invoices.
- Block or unblock users with predefined reasons.

#### Client Panel:

- o Place orders with menu search functionality.
- View total bill with GST.
- Cancel orders with reasons included in the invoice.

### • Order Management:

- Supports full and half portions (where applicable).
- Error handling for unavailable items.
- Automatic invoice generation and storage.
- Authentication System: Secure login and registration for all users.

# 3. System Requirements:

- Python: Version 3.8 or above
- Modules:
  - o os, json, datetime
  - o Install third-party modules (if any) using:

### bash

### Copy code:

- 1. pip install -r requirements.txt
- 2. pip install colorama maskpass tabulate

## 4.Installation:

1. Clone the Repository:

git clone https://github.com/your-username/Indixpert-RMS-Python-Pr03.git

2. Navigate to the Project Directory:

cd Indixpert-RMS-Python-Pr03

3. Install Dependencies:

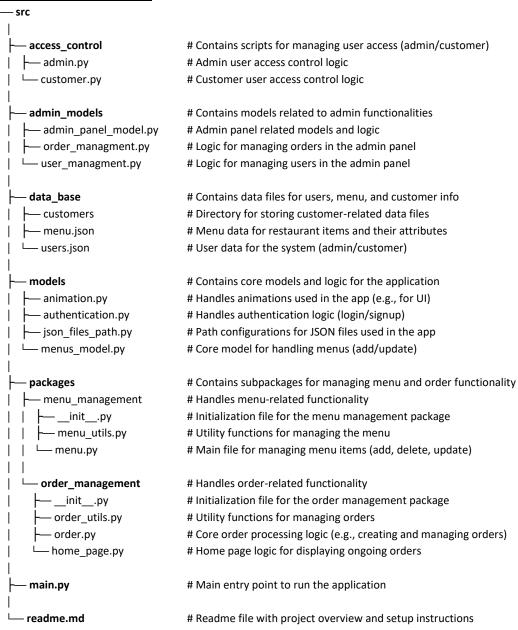
pip install -r requirements.txt

4. Run the Application:

python main.py

### **5.Project Structure:**

#### **INDIXPERT-RMS-PYTHON-PRO3**



## 6.<u>Usage</u>

### 1. Run the Application:

python main.py

### 2. Choose User Type:

o Admin: Full system access.

Client: Limited functionality.

### 3. Follow Prompts:

o Admin: Manage menu, view orders, block users, etc.

o Client: Place orders, view invoices, and reserve tables.

# 7. Business Logic

#### **Invoice Format**

Invoices are stored as JSON files in customer-specific directories and include:

- Order ID
- Customer details
- Item details (name, quantity, type)
- Subtotal, GST, and grand total
- Status (e.g., Completed, Canceled/Refunded)
- Cancellation reasons (if applicable)

### **Analytics Insights**

Analyzed data includes:

- Total revenue.
- Total refunds.
- Status-wise order distribution.
- Recommendations based on performance metrics

# 8. Classes and Functionalities

### 1. Menu Management (Menu Class):

- Add, update, delete, and display menu items.
- Store menu data in menu.json.

### 2. Order Management (Order Class):

- Take and manage orders.
- Includes search, availability checks, and pricing logic.

### 3. Billing System (Billing Class):

- Calculate subtotal, GST, and total amount.
- Generate and store invoices in JSON.

### 5. Admin & Client Panels:

• Separate classes to handle admin and client-specific workflows.

## 9. JSON Files and Their Roles

### 1. menu.json:

o Stores menu items with details like price, availability, and type.

### 2. invoices.json:

o Logs invoices for all orders placed.

## 3. users.json:

o Maintains user details, roles, and blocked status.

## 4. **Dynamic Folders**:

o Each customer has a separate folder for their invoices.

# **10.Future Enhancements**

- Implement a Graphical User Interface (GUI) using Tkinter or PyQt.
- Add payment gateway integration.
- Support multiple restaurants or branches.
- Enable **live updates** for menu changes.